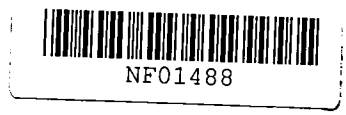
NASA Technical Memorandum 87170

Partitioning and Packing Mathematical Simulation Models for Calculation on Parallel Computers

Daie J. Arpasi and Edward J. Milner
Lewis Research Center
Cleveland, Ohio

Prepared for the
1986 International Conference on Parallel Processing
cosponsored by Pennsylvania State University and the IEEE Computer Society
St. Charles, Illinois, August 19–22, 1986



NASA

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA TM-87170		

4. Title and Subtitle	5. Report Date
Partitioning and Packing Mathematical Simulation Models for Calculation on Parallel Computers	6. Performing Organization Code: 505-62-3A

7. Author(s)	8. Performing Organization Report No.
Dale J. Arpasi and Edward J. Milner	E-2808
	10. Work Unit No.

9. Performing Organization Name and Address	11. Contract or Grant No.
National Aeronautics and Space Administration Lewis Research Center Cleveland, Ohio 44135	13. Type of Report and Period Covered: Technical Memorandum

12. Sponsoring Agency Name and Address	14. Sponsoring Agency Code
National Aeronautics and Space Administration Washington, D.C. 20546	

15. Supplementary Notes

Prepared for the 1986 International Conference on Parallel Processing, cosponsored by Pennsylvania State University and the IEEE Computer Society, St. Charles, Illinois, August 19-22, 1986.

16. Abstract

This paper deals with the development of multiprocessor simulations from a serial set of ordinary differential equations describing a physical system. Degrees of parallelism (i.e., coupling between the equations) and their impact on parallel processing are discussed. The problem of identifying computational parallelism within sets of closely coupled equations that require the exchange of current values of variables is described. A technique is presented for identifying this parallelism and for partitioning the equations for parallel solution on a multiprocessor. An algorithm which packs the equations into a minimum number of processors is also described. The results of the packing algorithm when applied to a turbojet engine model are presented in terms of processor utilization.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Simulation; Parallel processors	Unclassified - unlimited STAR Category 61

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of pages	22. Price*
Unclassified	Unclassified		

*For sale by the National Technical Information Service, Springfield, Virginia 22161

PARTITIONING AND PACKING MATHEMATICAL SIMULATION MODELS FOR CALCULATION ON PARALLEL COMPUTERS

Dale J. Arpasi and Edward J. Milner
National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135

SUMMARY

This paper deals with the development of multiprocessor simulations from a serial set of ordinary differential equations describing a physical system. Degrees of parallelism (i.e., coupling between the equations) and their impact on parallel processing are discussed. The problem of identifying computational parallelism within sets of closely coupled equations that require the exchange of current values of variables is described.

A technique is presented for identifying this parallelism and for partitioning the equations for parallel solution on a multiprocessor. An algorithm which packs the equations into a minimum number of processors is also described. The results of the packing algorithm when applied to a turbojet engine model are presented in terms of processor utilization.

INTRODUCTION

Multiple processors, operating together to solve a single problem, can, in many cases, decrease the time of calculation. This is important in time-critical applications, such as real-time simulation, where this technique can provide computational rates unachievable on a single processor or allow the use of lower cost hardware to provide the necessary computational capabilities. For certain classes of problems it is possible to configure a network of microcomputers to achieve the same throughput rate as a large mainframe computer at a lower initial and ongoing maintenance cost.

The parallel processing concept has opened new areas of research and development in hardware, software and theory. Some efforts sponsored by NASA Lewis are described in references 1 to 4. Techniques for developing mathematical models that can be solved efficiently on parallel processors is a key area of study. The first step in developing these multiprocessor models is to identify parallelism within the mathematical formulation of the problem. This requires a data flow analysis of the problem's equations and will identify the "critical path" and the minimum achievable calculation time. The next step is to arrange, or "pack" the noncritical path computations on the minimum number of processors so as to make maximum use of the available computing resources.

This paper presents a method of partitioning equations for multiprocessor solution. The method, when applied to models containing inherent parallelism, will produce segmented sets of equations that can be solved in parallel. A brief discussion of computational parallelism is presented in terms of the degree of coupling between subsets of the model equations. The identification of parallel paths in closely coupled sets of equations is then discussed, followed by the description of a packing algorithm for assignment of the paths

to a minimum number of processors. Finally, the results of the algorithm applied to the mathematical model of a helicopter engine are used to demonstrate the algorithm's capabilities.

COMPUTATIONAL PARALLELISM

A mathematical model of a physical system consists of a set of equations which describe, to some degree of accuracy, the response of that system to external influences (driving functions) over a limited range of operation. This range is defined in terms of the maximum and minimum values of the driving functions and, if time dependent, the maximum frequency or maximum rate of change of these functions. Generally, the object of this modeling effort is to provide a simulation of the physical system.

Whenever a simulation must interface to the real world (e.g., with control hardware), the mathematical model must be computed at a rate sufficient to make the simulation look like the real physical process to the real-world components. This is termed real-time computation. To test a piece of control hardware, the simulation response to control inputs must dynamically match the response of the physical system. This real-time requirement establishes a maximum allowable calculation time for a digital simulation. The maximum allowable calculation time may be further reduced by internal numerical stability requirements. The maximum calculation time necessary to meet all of these real-time requirements is termed the required update time (ΔT). The simulation will meet all real-time requirements if the mathematical model is computed every ΔT sec.

Once the simulation update time has been established, the model must be programmed and executed on a computer. Both the efficiency of the programming language and the capabilities of the computer hardware will determine the achievable calculation time of the model. If the model can be divided into parts which can be computed in parallel, then the efficiency requirements of the language and the computational capabilities of the hardware may be relaxed by assigning each path to a separate computer and computing these paths in parallel (assuming appropriate data transfer takes place between the computers.) If a model can be computed serially in the required update time on a single computer, parallel processing may allow the use of a number of lower cost computers to do the job. This could be a more cost effective approach to simulation.

The prerequisite to developing parallel processor simulations is to be able to identify the parallel computational paths contained in the model. In general, a dynamic model can be programmed on a digital computer as a set of N computationally sequential equations of the form

$$X_K\,(ih) = f_K[X_m(ih),\ X_m((i-1)h),\ \ldots,\ u(ih)]$$

where $X_K(ih)$ is the result of the K^{th} equation at time ih. Here, h denotes the simulation time step or update interval of the model calculations. The arguments $X_m(ih)$ are the current values of the results of preceding equations in the model (i.e., $m = 1$ to $K - 1$), and $X_m((i-1)h)$ are the past values of the results of all equations in the model. The argument u represents values obtained from sources external to the model which are always available at the start of the model computation sequence. The functional

relationship between X_K and its arguments is represented by f_K. Assuming an equation is an indivisible computational unit, then the parallelism in the model is determined by the arguments of each equation. That is, two equations, or sets of equations, can be computed in parallel within an update interval if their arguments are independent of the results of the others computed in that interval.

For example, a model of the form

$$X_1(ih) = f_1[X_3((i - 1)h)]$$

$$X_2(ih) = f_2[X_1((ih))]$$

$$X_3(ih) = f_3[X_2((ih)), X_3((i - 1)h)]$$

contains no parallelism since $X_3(ih)$ requires $X_2(ih)$ and $X_2(ih)$ requires $X_1(ih)$. These calculations must be done serially. However, the model

$$X_1(ih) = f_1[X_3((i - 1)h)]$$

$$X_2(ih) = f_2[u(ih)]$$

$$X_3(ih) = f_3[X_1(ih), X_2(ih), X_3((i - 1)h]$$

does contain parallelism since X_1 can be computed at the same time as X_2.

Many times, parallelism is obvious from the physical nature of the system which is modeled. For example, to test a control system designed to balance the load on two generators, the generators could be formulated in a model containing two sets of decoupled equations. That is, there are no result values transferred between the sets. Both, however, provide results to an external piece of hardware (the controller). It is obvious that the simulation of the generators can at least be implemented on two computers operating in parallel (fig. 1(a)). Each equation set (generator) may, of course, contain additional computational parallelism.

Another example of physically detectable parallelism occurs when two or more sets of equations have significantly different dynamic characteristics. For example, suppose a simulation of a single engine aircraft is desired. The mathematical model would consist of a set of engine equations and a set of airframe equations. The airframe dynamics are generally an order of magnitude slower than the engine dynamics. Therefore, these equation sets can be computed using different update times. If the update times differ by a factor of 5 or more (rule of thumb), then results transferred between the airframe and engine can be past values without introducing dynamic errors. In this case, the equation sets are considered loosely coupled and each can be placed on a separate computer. As shown in figure 1(b), because the airframe equations (A) use only results of engine equations (E) computed in a previous calculation (update interval) and vice-versa, these arguments can be computed in parallel. As is the case of decoupled equation sets, loosely coupled equation sets may each contain additional computational parallelism.

Parallelism due to decoupled or loosely coupled equation sets is easily identified from the physical nature of the model. A more difficult task is the identification of parallelism in a set of closely coupled equations, where the process dynamics dictate the use of current arguments in solving for equation results. For instance, suppose a model contains the following set of equations:

$$X_1(ih) = f_1[X_5((i-1)h),\ u(ih)]$$

$$X_2(ih) = f_2[X_5((i-1)h)]$$

$$X_3(ih) = f_3[X_2(ih)]$$

$$X_4(ih) = f_4[X_1(ih),\ X_2(ih)]$$

$$X_5(ih) = f_5[X_3(ih),\ X_4(ih),\ X_5((i-1)h)]$$

The variable X_1 can be computed at the start of the calculation interval, since it is a function of the past value of X_5 and the external variable u. X_2 may also be computed at the start of the interval. However, the calculation of X_3 must be delayed until X_2 has been determined, the calculation of X_4 must be delayed until both X_1 and X_2 are determined, and the calculation of X_5 must be delayed until both X_3 and X_4 have been determined. As shown in figure 2(a), three computational paths can be identified which can be assigned to three different computers in the simulation. Note that "wait states" have been inserted to insure the currency of the arguments. That is, equation calculation is delayed until current argument values become available. The X_3 calculation is shown to be delayed slightly for the transfer of X_2. The shaded areas (time slots) indicate the time available for result transfer to computer number 1. The calculation of X_1 and X_3 can take place anywhere in the time slot.

The detection of this type of computational parallelism can become burdensome when the equation set becomes large. The technique, however, can be automated. Related to this problem of partitioning is the problem of allocation (i.e., packing these paths into a minimum number of computers without extending the update time). Figure 2(b) demonstrates packing of the paths defined in figure 2(a). Arbitrary calculation times of TX_1, TX_2, TX_3, TX_4 and TX_5 have been assigned to the equations producing results X_1 through X_5, respectively. The time TX_1 includes the time required to obtain the value of u. Note in figure 2(b), that, because of the calculation times, the $X_2 - X_4 - X_5$ path is critical in that it contains no idle states. This path, therefore, dictates the minimum possible update time ($\Delta T = TX_2 + TX_4 + TX_5$). The paths X_1 and X_3 are assigned to separate computers. Packing in this example is a trivial task, since the X_3 calculation can be moved onto the computer number 2 to be calculated during the idle period.

In many cases, efficient packing requires shifting equations in their time slots. This causes a ripple effect on the time slots of other equations which can complicate the packing problem. Because of the nature of the packing problem, a unique solution to the development of a packing algorithm does not exist. There are many ways to pack most parallel models. The optimum approach may be model-dependent.

In the following sections partitioning and packing algorithms that have been developed at NASA Lewis are discussed. These algorithms were tested with a model of a jet engine and the results are presented.

PARTITIONING

To begin the discussion of the partitioning algorithm, certain terms should be defined. A mathematical model is a set of equations, written to define the characteristics of a physical system to some desired degree of accuracy. A program is a sequential set of digital equations and supporting information (e.g., variable and constant definition) which define the mathematical model within the constructs of a programming language. A path is a subset of these equations which, because of interrelationships between arguments and results, contains no parallelism. Partitioning is the transformation of the program equations into a number of paths which may be calculated in parallel. Packing is the combination of paths into a minimum number of processors (computers) which, provide computation of the model within a prescribed update interval. The critical path is the longest path and the prescribed update interval must be greater than or equal to the calculation time of the critical path.

In this discussion of partitioning it is assumed that a program is given. That is, these equations, when executed serially, provide the required results. No assumptions are made concerning the parallelism of computational units (operations) contained in the program equations. The equation

$$x = a*y + b*z$$

contains parallelism (i.e., a*y can be calculated in parallel with b*z) which will be ignored since we are concerned with partitioning at the equation level and not parsing. For purposes of this discussion, the above equation will be considered as

$$x = f(a, y, b, z)$$

where f is some single operation. Therefore, equations will be assigned to paths in their entirely and not broken up into more primitive result-argument relationships.

As indicated in the last section, partitioning requires the establishment of result-argument relationships for the serial set of equations in order to develop computational paths. It is also necessary to know the calculation time of each equation. The program must be processed to provide this information. For this effort, the result-argument relationships and the calculation time information are outputs of the multiprocessor programming utility RTMPL (refs. 2 and 3). The primary function of this utility is to translate a structured program of the mathematical model into assembly language for the simulation processor(s). As an option, the utility also provides information on the result and arguments of each equation, the processor operations necessary to obtain the result, and the processor calculation time for each operation. For example, the utility-generated information for the equation

X = y + z

might be as shown in table I. Each equation has a label (programmer or RTMPL defined). In this case, S$12 is used to indicate that this information concerns the twelfth equation in the program. The processor operations to compute the equation are: load register R1 with z (requires 8 time units), add variable y to R1 (16 time units) and store R1 as the value of variable X (8 time units). This type of information is generated for each equation in the program.

The first step in the partitioning process is to convert the utility generated information into the form needed for partitioning. This form is shown in table II for the close-coupled example in the previous section. To simplify the discussion, the equation label has been replaced by the name of the result variable. Dependent arguments are those which are the results of previous program equations calculations in the update interval (e.g., X_1 is a dependent argument of X_4). These are the drivers for partitioning since their current values are required before the computation sequence can continue. The independent arguments u and X_5 do not affect partitioning since only past values are used. The calculation time for each equation is determined by adding the calculation times of the given processor operations. For example, the calculation time of equation X_1 is determined to be 32 from table I.

The time at which an equation can start is determined by the arguments and calculation time of each equation. The first equation of a set only has independent arguments and thus, can always start at time 0 (measured from the beginning of the calculation update interval). It can never require results from calculations in the current update interval since none are yet available. An equation can end at the time obtained by adding its calculation time to the time it can start. The general formula for obtaining this time is

CANSTART(RESULT) = MAX(CANEND(ARG 1), CANEND(ARG 2), ..., 0)

CANEND(RESULT) = CANSTART(RESULT) + CALCTIME(RESULT)

where ARG1 is the first dependent argument, etc. This formula is applied sequentially to each equation in the program (see table II).

Once these attributes have been established for each program equation, the identification of computational paths contained in the program can begin. The algorithm used for path identification is shown in figure 3. Its purpose is to identify all sequences of equations which contain no parallelism and which must be computed serially. These paths are organized into a linked list called PATHLIST. The paths in PATHLIST are ordered in terms of decreasing path calculation time. Therefore. the first path in PATHLIST is the critical path. To form a path, the algorithm selects the program equation, having the maximum CANEND time, and which has not already been assigned to a path. This is the result equation of the path. The next equation selected is the one which produces a result used as a dependent argument of the result equation. If more than one equation result is used as a dependent argument, then the one with maximum CANEND time is selected. The selected equation then is inserted in front of the result equation in the path. The path formation process continues until an equation is inserted which has no dependent arguments equations which are not already assigned to a path. Paths are formed until all program equations have been assigned.

Partitioning has been discussed in terms of equations that produce values of variables. Often, mathematical models contain statements that do not produce values. Two common examples are conditional statements (e.g., IF ... THEN ... ELSE) and command statements (e.g., I/O operations). The calculation time of such a statement must be combined with a preceeding or following equation. This could impose limitations on program structure and is a subject for future study.

PACKING

The partitioning process produces a number of paths consisting of equations which must be computed serially and a table of information on each equation (described in table II). The final task in the process of formulating a multiprocessor model is to pack the paths for assignment to a minimum number of processors. The first path in our list has the largest calculation time due to the partitioning algorithm. This is called the critical path and its calculation time is the minimum time in which the model can be computed no matter how many processors are used. The number of paths identified through partitioning is usually greater than the minimum number of processors necessary.

The minimum number of processers necessary to implement a multiprocessor simulation depends on how fast the simulation must be computed. This update time must be specified prior to packing. The simulation time step h is usually based on stability and dynamic accuracy requirements. For real-time applications, the update time ΔT must be equal to h. The update time also specifies when the computations must end. The first step in packing is to determine when each equation must end using the specified update time. By doing so the last column of information is added to table II.

To determine when an equation must end, we begin with the state variables (defined here as those variables whose current values are not used as arguments in the model, but appear as results of model equations). The state variable computations will be the last computations performed, and thus must will end at the prescribed update intervals. The calculation of equations which are dependent arguments of these variables must end no later than the time at which the state variable calculations must start. The times when subsequent equations in the result/argument string must end is similarly determined. Since a variable can be used as a dependent argument in more than one equation, care must be taken that the earliest time, arrived at after all paths are analyzed, is used to specify when that equation must end. The numbers given in table II were arrived at by specifying an update time equal to the time at which the state variable equation X_5 can end. Note that both X_3 and X_4 use X_2 as a dependent argument. The time at which X_2 must end, as determined from X_3 requirements, is

$$\text{MUSTEND}(X_2) = 112 - 32 = 80$$

and, from X_4, is

$$\text{MUSTEND}(X_2) = 112 - 48 = 64$$

The minimum is selected.

We now have determined when an equation can start, can end, and must end. These are termed equation attributes. Since the paths are serial they can also be assigned these attributes: A path can start when its first equation can start, a path can end when its last equation can end, a path must end when its last equation must end, and additionally, the calculation time of a path is the summation of the calculation times of its equations. This is sufficient information to pack the paths.

The solution to the packing problem is not unique in that many arrangements of paths in processors can result in a satisfactory solution. The requirements placed on a general packing algorithm are not strict, however, from the point of view of efficiency of processor utilization. For example, it does not matter if the last processor that is packed performs its calculations in 5 or 95 percent of the update time if there is insufficient time to calculate all of its equations on the other processors used in the simulation. From a processor utilization point of view, both packing arrangements are satisfactory since the same number of processors are used in the simulation.

The packing algorithm, shown in figure 4, was designed to achieve the minimum number of processors. Other requirements which may be imposed, such as memory size limitations and inter-processor data transfer limitations were not imposed on the algorithm.

As input, the algorithm requires: 1) that all paths be specified in a linked list called PATHLIST in order of decreasing calculation time; 2) that the required update time of the simulation, ΔT, is specified and that the attributes of each equation and path (CANSTART, CANEND, MUSTEND, CALCTIME) have been determined as described above.

The packing algorithm creates processors as needed and inserts paths from PATHLIST according to a hierarchy of relationships between existing equations in a processor and the equations in the unpacked paths. When a processor is created, the path with the longest calculation time in PATHLIST is inserted. Next, the paths which are related to paths already in a processor are tested to see if they fit (see discussion of TESTFIT algorithm below). If so, they are inserted, if not, they are placed in a carry-over list.

Then, paths in PATHLIST which are unrelated to the equations in the processor are tested. If one of these is inserted, unrelational testing is ended and relational testing begins again. When no other paths can be inserted into a processor, another processor is created. This process continues until all paths in PATHLIST are inserted into a processor.

Relational testing is prioritized. All unpacked paths which provide critical arguments are tested first. (A path is considered to provide a critical argument if the result of the last equation in the path (EL) is an argument of a processor equation (EP) and

$$\text{MUSTEND (EL)} = \text{MUSTEND (EP)} - \text{CALCTIME (EP)}$$

Next, other related paths are tested. Then paths in the carry over list (which was formed from paths which were related to equations packed into previously formed processors, but not yet packed) are tested.

Paths are tested for insertion on an equation by equation basis using the test fit algorithm shown in figure 5. First the attributes (CANSTART, CANEND, CALCTIME, and MUSTEND) of all program equations are saved. This is necessary because inserting an equation into a processor can cause a ripple effect on the attributes of other equations. If the whole path does not fit, any equation of the path, inserted into the processor, must be removed and the attributes of affected equations restored.

The ripple effect is illustrated in figure 6. Assume a processor contains two equations (A and B) and that it has been determined that equation (C) can be inserted between them. The calculation time of each equation is shown as the shaded areas. For packing purposes, the calculation of each equation can take place anytime between its CANSTART time and its MUSTEND time.

The space available for C is the difference between the time at which B must end and A can end minus the calculation time of B. Equation C will be inserted to start directly after A can end. The calculation of B will be delayed until C can end. Note that the difference between the MUSTEND and CANEND times of A and B have been reduced to zero by the positioning of C, and that the time difference for C has been reduced. The primary impact of these changes is to reduce the space in the processor available for packing other paths. There is also a secondary impact of equal importance. By increasing the times at which C and B can end, any unpacked equations which use these equations as arguments have their starting times delayed. Similarly, by reducing the MUSTEND times of A and C, the MUSTEND times of any unpacked arguments of these equations are moved up. These effects tend to reduce the slot sizes of unpacked equations restricting the range of time into which they can be packed into a processor. Also, these ripple effects may introduce computational gaps within unpacked paths.

After the attributes are saved (fig. 5), the path equations are ordered in terms of decreasing CANEND times for insertion testing. That is, the latest equation will be tested first and the earliest last.

The processor equations are arranged in sequential order where EP(1) is the earliest equation and EP(n) is the latest equation. Testing to determine if a path equation (E(i)) can be inserted into the processor involves the identification of all slots between any two processor equations (EP(j - 1), EP(j)) where the equation fits. The processor end points (i.e., EP(j) = EP(1) and EP(j - 1) = EP(n)) must also be considered. Because of the argument and result relationships between E(i) and the processor equations it is required that the range of processor equations be limited for testing purposes. Let the end points of the range be designated by EPE and EPL (the earliest and latest processor equations respectively, before which E(i) may be inserted). This range is established as follows: If E(i) is an argument of a processor equation, then EPL is the earliest processor equation of which E(i) is an argument (EPE = EP(1)); if any processor equation is an argument of E(i), then EPE is the one following the latest of these and EPL is the last processor equation plus one (end point); if E(i) is unrelated to any processor equation, then EPE = EP(1) and EPL is the last processor equation plus one.

Once the range of testing has been established, all slots within that range are tested to determine if E(i) fits. The fit criterion is as follows:

1. If EP(j) = EP(1) then CANEND* (E(i)) = CALCTIME (E(i)) else CANEND* (E(i)) = CANEND (EP(j - 1)) + CALCTIME (E(i));

2. If CANEND* (E(i)) < CANEND (E(i)) then CANEND* (E(i)) = CANEND (E(i));

3. If EP(j - 1) = EP(n) then MUSTEND* (E(i)) = ΔT else MUSTEND* (E(i)) = MUSTEND (EP(j)) - CALCTIME (EP(j));

4. If MUSTEND* (E(i)) > MUSTEND (E(i)) then MUSTEND* (E(i)) = MUSTEND (E(i))

5. IF [[CANEND* (E(i)) < = MUSTEND (E(i))] and [CANEND* (E(i)) < = MUSTEND* (E(i))]] then E(i) fits.

The asterisk indicates the attributes of E(i) if it were inserted into the slot between EP(j - 1) and EP(j).

If it is established that E(i) can fit in more than one slot, the test-fit algorithm proceeds to select the slot into which E(i) best fits. The best fit criterion is as follows:

If a slot exists such that

$$\text{CANEND*}\ (E(i)) - \text{CALCTIME}\ (E(i)) - \text{CANEND}\ (EP(j-1)) = 0$$

then this slot is selected. Otherwise, the latest slot which maximizes

$$[\text{MUSTEND*}\ (E(i)) - \text{CANEND*}\ (E(i))]$$

is selected.

This criterion provides for efficient packing by eliminating processor idle time if possible, and if not, then the ripple effect from insertion is minimized.

Once a slot is selected, the equation is inserted and the attributes of all program equations are updated to reflect the insertion. If any path equation cannot be inserted into the processor, path equations which have already been inserted are removed from the processor, the original attributes are restored to the program equations and the Test Fit algorithm ends.

RESULTS

The packing algorithm was programmed in Pascal, along with the partitioning algorithm. It was then tested on a helicopter engine model. The appendix contains a complete listing and description of the output. The results, in terms of percent processor utilization, are presented in table III. This first column is the update time ΔT specified prior to packing. It is given in terms of machine cycles. The second column gives the number of processors required for packing. The remaining columns show the percent utilization of the update time in calculating each processor's assigned equations. The first specified update time (5666 cycles) was the minimum possible time and corresponds to the critical path calculation time. In this case, four processors

were required. The second update time selection (10 000 cycles) required two processors. Note that the percent utilization of the last processor in each case exceeds the summation of time available on the other processor(s). The algorithm, therefore, functioned satisfactorily.

Since the packing algorithm does not account for data transfer times, it is possible that the available time between when a variable is computed on one processor and when its value is required for computation on another processor will be less than the time required to transfer the variable between the processors. The effect of this will be to increase the effective calculation time of the packed simulation, and therefore, to increase the minimum achievable update time. Data transfer effects may be significant for multiprocessor systems with inefficient data transfer mechanisms or for simulations that require large volumes of data transfer between processors. Future work in the development of a packing algorithm should include a study of the effects of data transfer. While these effects will increase the critical path time, and therefore, the minimum update time, proper consideration of data transfer will minimize this increase and provide for more efficient packing.

CONCLUDING REMARKS

The algorithms and considerations presented for partitioning and packing mathematical models for calculation on parallel processors has simplified the development of multiprocessor simulations at NASA Lewis. Evaluation of the packing algorithm will continue as other multiprocessor simulations are developed. Work on a completely automated programming package is in progress, which will take a structured serial statement of a mathematical model, detect its parallelism, and provide load modules for the required number of processors.

The authors welcome discussions of the techniques presented in the paper, related techniques, and developments in the many other aspects of multiprocessor simulation.

APPENDIX

The following listing is a result of partitioning and packing the program equations describing a helicopter engine. An update interval equal to the critical path time was specified. All times are given in terms of machine cycle time of the Motorola 68000 microprocessor.

Listing pages 1 to 3 list the program equations in terms of result variables. For each equation, the calculation time (EXTIME) and the CANSTART and MUSTEND attributes are given. The "RELATED EQUATION" column identifies the relationships between the equation and other equations. If the equation is an argument of other equations it is so indicated by an "UB" entry. Constant arguments of the equation are designated by "CN" followed by the name of the constant. An "SV" entry designates those equations whose past values are arguments of the equation. Current value or dependent argument equations are indicated by a "RQ" entry.

The paths generated by the partitioning algorithm are shown at the top of listing page 4. Nineteen paths were found. The attributes of each path are given, followed by the equations contained in each path. Path number 1 is the critical path.

The packing sequence beginning in the middle of listing page 4 and ending on page 11 is an optional diagnostic listing which can be selected by the user. It details the operation of the packing algorithm in processing the paths. Each step is preceded by asterisks. The step is then defined and the operable equations identified. The number preceding the equation list is the appropriate path number, if applicable. The number zero implies that the equations are contained in a temporary working list. The last comment on listing page 11 indicates whether or not the processors were successfully packed. If the smallest path (processor) execution time is less than the unused time on the other paths (processors) then the packing is successful.

Listing pages 12 to 14 define the equations packed into each processor. The processors are identified by number (in this case 1 to 4) and the percent utilization of the prescribed update interval is given. The calculation time (CALC), the time (relative to the start of the update interval) at which the equation starts and ends its computation, and the time computation must end (MAX) in order to meet data transfer requirements is listed. These requirements are also listed for each equation. The processors ("PR") and their equations which use an equation as a dependent argument are shown in the "SENT TO" column. The time at which the value is required to arrive at the destination processor (REQ) is also provided. The "NEEDS" column indicates those processors and equations which supply dependent argument values to an equation. The time at which these values are available (AVAIL) is also included.

EQUATION	EXTIME	CANSTRT	MUSTEND	RELATED EQUATIONS
DEL2	218	0	3856	UB: WA2
				CN: P2
RTTH2	770	0	770	UB: WA2
				CN: T2
WF	214	0	2316	UB: FAR41
				CN: WFPH
P3	242	0	1076	UB: PS3
				SV: T3-WS3
PS3	122	242	1198	RQ: P3
PS3Q2	204	364	1402	RQ: PS3
				UB: WA2C-T25Q2
				CN: P2
T3Q2	598	568	4832	RQ: PS3Q2
				CN: XPRC1-NPRC1-ZTRC
T3C	134	1166	4966	RQ: T3Q2
				CN: T2
NGC	434	770	1204	RQ: RTTH2
				SV: NG
PCNGC	198	1204	1402	RQ: NGC
				UB: B1
WA2C	2136	1402	3538	RQ: PS3Q2-PCNGC
				UB: WXQ2-WA2
				CN: XPRC-YPNGC-NPRC-ZW2C
WA2	292	3538	4148	RQ: WA2C-DEL2-RTTH2
				UB: WB25-WA3-TORQC-P45DT-WB3
B1	598	1402	4148	RQ: PCNGC
				UB: WB25
				CN: XPNGC-NPNGC-ZB1
B2	610	3538	4148	RQ: WA2C
				CN: XW2CB-NW2CB-ZB2
WB25	122	4148	4270	RQ: B1-B2-WA2
WA3	30	4270	4300	RQ: WA2-WB25
				UB: WS3DT
WXQ2	598	3538	4990	RQ: WA2C
				UB: WB3
				CN: XW2C-NW2C-ZWXQ2
WB3	130	4136	5330	RQ: WXQ2-WA2
				UB: WS3DT
WA31	970	242	2316	RQ: P3
				UB: P41DT-WS3DT
				SV: P41-WS3
H3	142	0	2540	UB: TORQC-H41
				SV: T3
FAR41	224	1212	2540	RQ: WF-WA31
H41	336	1436	2876	RQ: FAR41-H3
T41	122	1772	2998	RQ: H41
				UB: P41DT
THTA41	122	1894	3120	RQ: T41
				UB: W41
W41	838	2016	4504	RQ: THTA41
				UB: P45DT-TORQ41
				SV: P41
PR45Q1	230	0	2522	SV: P41-P45
HGTH4	598	230	3120	RQ: PR45Q1

```
                                                UB: DH41
                                                CN: XPRGT-NPRGT-ZDHGTQ
 A41          118       2016       3238        RQ: DHQTH4-THTA41
                                                UB: TORQ41
TORQ41        390       2854       4894        RQ: W41-DH41
                                                UB: NGDT
                                                SV: NG
T25Q2         134        568       4068        RQ: PS3Q2
T25           118        702       4186        RQ: T25Q2
                                                CN: T2
H25           114        820       4300        RQ: T25
                                                UB: TORQC
H2            126          0       4300        UB: TORQC
                                                CN: T2
TORQC         594       4300       4894        RQ: H25-WB25-H2-WA2-H3-WA3
                                                SV: NG
H44            54       2134       3292        RQ: H41-DH41
H45           114       2188       3406        RQ: H44
                                                UB: H49
T45           122       2302       3528        RQ: H45
                                                UB: P45DT
PR49Q5        230          0       3052        UB: W45C
                                                CN: P49
                                                SV: P45
W45C          598        230       3650        RQ: PR49Q5
                                                UB: W45
                                                CN: XPRPT-NPRPT-ZW45C
THTA45        122       2424       3650        RQ: T45
                                                UB: DH45
W45           854       2546       4504        RQ: THTA45-W45C
                                                UB: P45DT
                                                SV: P45
DHQTH5        610        230       4386        RQ: PR49Q5
                                                UB: DH45
                                                CN: XPRPT-NPRPT-ZDHPTQ
DH45          118       2546       4504        RQ: DHQTH5-THTA45
                                                UB: TORQ45
TORQ45        390       3400       4894        RQ: W45-DH45
                                                SV: NP
H49            42       2664       5666        RQ: H45-DH45
NGDT          566       4894       5460        RQ: TORQ41-TORQC
NPDT          566       3790       5460        RQ: TORQ45
                                                CN: TORQLD
P41DT         312       2854       5460        RQ: W41-WA31-T41
P45DT         470       4136       5460        RQ: WXQ2-WA2-W45-W41-T45
T3DT          494       1300       5460        RQ: T3C
                                                SV: T3
WS3DT         130       4300       5460        RQ: WA3-WA31-WB3
NG            206       5460       5666        RQ: NGDT
                                                CN: DELTAT
                                                SV: NG
NP            206       4356       5666        RQ: NPDT
                                                CN: DELTAT
                                                SV: NP
 41           206       3166       5666        RQ: P41DT
                                                CN: DELTAT
```

```
                                                SV: P41
P45             206     4606    5666            RQ: P45DT
                                                CN: DELTAT
                                                SV: P45
T3              206     1794    5666            RQ: T3DT
                                                CN: DELTAT
                                                SV: T3
WS3             206     4430    5666            RQ: WS3DT
                                                CN: DELTAT
                                                SV: WS3
```

P#	EXTIME	CANSTRT	MUSTEND	COMPUTATIONAL PATHS
1	5666	0	5666	RTTH2-NGC-PCNGC-WA2C-B2-WB25-WA3-TORQC-NGDT-NG
2	4562	0	5666	P3-WA31-FAR41-H41-T41-THTA41-DH41-H44-H45-T45-THTA45-W45-TORQ45-NPDT-NP
3	1758	242	5666	PS3-PS3Q2-T3Q2-T3C-T3DT-T3
4	1356	2016	5666	W41-P41DT-P41
5	1274	3538	5666	WXQ2-P45DT-P45
6	840	0	4386	PR49Q5-DHQTH5
7	828	0	3120	PR45Q1-DHQTH4
8	598	1402	4148	B1
9	598	230	3650	W45C
10	390	2854	4894	TORQ41
11	366	568	4300	T25Q2-T25-H25
12	336	4300	5666	WS3DT-WS3
13	292	3538	4148	WA2
14	218	0	3856	DEL2
15	214	0	2316	WF
16	160	2546	5666	DH45-H49
17	142	0	2540	H3
18	130	4136	5330	WB3
19	126	0	4300	H2

PACKING SEQUENCE

```
*****  PACKING PROCESSOR 1
  1. RTTH2-NGC-PCNGC-WA2C-B2-WB25-WA3-TORQC-NGDT-NG
*****  RELATIONAL EQUATIONS ARE:
  0. PS3Q2-B1-WA2-H25-H2-H3-TORQ41-WXQ2-WS3DT
*****  PACKING RELATIONAL EQUATIONS
*****  ORDERED EQUATIONS TO BE PACKED:
  0. PS3Q2-WXQ2-B1-TORQ41-H25-WS3DT-WA2-H3-H2
*****  NOT PATH ENDPOINT:PS3Q2
*****  ORDERED EQUATIONS TO BE PACKED:
  0. WXQ2-B1-TORQ41-H25-WS3DT-WA2-H3-H2
*****  TRYING TO INSERT PATH 5
  5. WXQ2-P45DT-P45
*****  PATH DOES NOT FIT, ADDED TO CARRYOVER FOR NEXT PROCESSOR
*****  ORDERED EQUATIONS TO BE PACKED:
  0. B1-TORQ41-H25-WS3DT-WA2-H3-H2
*****  TRYING TO INSERT PATH 8
  8. B1
*****  PATH DOES NOT FIT, ADDED TO CARRYOVER FOR NEXT PROCESSOR
*****  ORDERED EQUATIONS TO BE PACKED:
  0. TORQ41-H25-WS3DT-WA2-H3-H2
*****  TRYING TO INSERT PATH 10
 10. TORQ41
*****  PATH DOES NOT FIT, ADDED TO CARRYOVER FOR NEXT PROCESSOR
*****  ORDERED EQUATIONS TO BE PACKED:
  0. H25-WS3DT-WA2-H3-H2
*****  TRYING TO INSERT PATH 11
 11. H25-T25-T25Q2
 *****  PATH DOES NOT FIT, ADDED TO CARRYOVER FOR NEXT PROCESSOR
*****  ORDERED EQUATIONS TO BE PACKED:
```

```
  0. WS3DT-WA2-H3-H2
***** TRYING TO INSERT PATH 12
 12. WS3DT-WS3
***** PATH DOES NOT FIT, ADDED TO CARRYOVER FOR NEXT PROCESSOR
***** ORDERED EQUATIONS TO BE PACKED:
  0. WA2-H3-H2
***** TRYING TO INSERT PATH 13
 13. WA2
***** PATH DOES NOT FIT, ADDED TO CARRYOVER FOR NEXT PROCESSOR
***** ORDERED EQUATIONS TO BE PACKED:
  0. H3-H2
***** PACKING DELAYED, NOT CRITICAL:H3
***** ORDERED EQUATIONS TO BE PACKED:
  0. H2
***** TRYING TO INSERT PATH 19
 19. H2
***** PATH DOES NOT FIT, ADDED TO CARRYOVER FOR NEXT PROCESSOR
***** ORDERED EQUATIONS TO BE PACKED:
  0. H3
***** TRYING TO INSERT PATH 17
 17. H3
***** PATH DOES NOT FIT, ADDED TO CARRYOVER FOR NEXT PROCESSOR

***** PACKING PROCESSOR 2
  2. P3-WA31-FAR41-H41-T41-THTA41-DH41-H44-H45-T45-THTA45-W45-TORQ45-NPDT
     -NP
***** RELATIONAL EQUATIONS ARE:
  0. WF-H3-DHQTH4-W45C-DH45-PS3-W41-TORQ41-H49-P41DT-P45DT-WS3DT
 **** CARRYOVER EQUATIONS ARE:
  0. H3-H2-WA2-WS3DT-H25-TORQ41-B1-WXQ2
***** PACKING RELATIONAL EQUATIONS
***** ORDERED EQUATIONS TO BE PACKED:
  0. PS3-W41-P41DT-P45DT-DHQTH4-W45C-TORQ41-WS3DT-WF-DH45-H49-H3
***** TRYING TO INSERT PATH 3
  3. PS3-PS3Q2-T3Q2-T3C-T3DT-T3
  2. P3-PS3-WA31-FAR41-H41-T41-THTA41-DH41-H44-H45-T45-THTA45-W45-TORQ45
     -NPDT-NP
  2. P3-PS3-PS3Q2-WA31-FAR41-H41-T41-THTA41-DH41-H44-H45-T45-THTA45-W45-TORQ45
     -NPDT-NP
  2. P3-PS3-PS3Q2-WA31-FAR41-H41-T41-THTA41-DH41-H44-H45-T45-THTA45-W45-T3Q2
     -TORQ45-NPDT-NP
  2. P3-PS3-PS3Q2-WA31-FAR41-H41-T41-THTA41-DH41-H44-H45-T45-THTA45-W45-T3Q2
     -TORQ45-T3C-NPDT-NP
***** PATH DOES NOT FIT, ADDED TO CARRYOVER FOR NEXT PROCESSOR
***** ORDERED EQUATIONS TO BE PACKED:
  0. W41-P41DT-P45DT-DHQTH4-W45C-TORQ41-WS3DT-WF-DH45-H49-H3
***** TRYING TO INSERT PATH 4
  4. W41-P41DT-P41
  2. P3-WA31-FAR41-H41-T41-THTA41-DH41-H44-H45-T45-THTA45-W45-W41-TORQ45
     -NPDT-NP
  2. P3-WA31-FAR41-H41-T41-THTA41-DH41-H44-H45-T45-THTA45-W45-W41-TORQ45
     -NPDT-NP-P41
***** PATH DOES NOT FIT, ADDED TO CARRYOVER FOR NEXT PROCESSOR
***** ORDERED EQUATIONS TO BE PACKED:
  0. P41DT-P45DT-DHQTH4-W45C-TORQ41-WS3DT-WF-DH45-H49-H3
***** NOT PATH ENDPOINT:P41DT
```

```
***** ORDERED EQUATIONS TO BE PACKED:
  0. P45DT-DHQTH4-W45C-TORQ41-WS3DT-WF-DH45-H49-H3
 **** NOT PATH ENDPOINT:P45DT
***** ORDERED EQUATIONS TO BE PACKED:
  0. DHQTH4-W45C-TORQ41-WS3DT-WF-DH45-H49-H3
***** TRYING TO INSERT PATH 7
  7. DHQTH4-PR45Q1
  2. P3-WA31-FAR41-H41-T41-THTA41-DHQTH4-DH41-H44-H45-T45-THTA45-W45-TORQ45
     -NPDT-NP
  2. P3-WA31-FAR41-H41-T41-THTA41-PR45Q1-DHQTH4-DH41-H44-H45-T45-THTA45-W45
     -TORQ45-NPDT-NP
***** PATH ADDED
***** UNPACKED PATHS: 3.T3;4.P41;5.P45;6.DHQTH5;8.B1
  9.W45C;10.TORQ41;11.H25;12.WS3;13.WA2;14.DEL2;15.WF;16.H49
  17.H3;18.WB3;19.H2
***** ORDERED EQUATIONS TO BE PACKED:
  0. W45C-TORQ41-WS3DT-WF-DH45-H49-H3
***** TRYING TO INSERT PATH 9
  9. W45C
***** PATH DOES NOT FIT, ADDED TO CARRYOVER FOR NEXT PROCESSOR
***** ORDERED EQUATIONS TO BE PACKED:
  0. TORQ41-WS3DT-WF-DH45-H49-H3
***** TRYING TO INSERT PATH 10
 10. TORQ41
***** PATH DOES NOT FIT, ADDED TO CARRYOVER FOR NEXT PROCESSOR
***** ORDERED EQUATIONS TO BE PACKED:
  0. WS3DT-WF-DH45-H49-H3
***** TRYING TO INSERT PATH 12
 12. WS3DT-WS3
  2. P3-WA31-FAR41-H41-T41-THTA41-PR45Q1-DHQTH4-DH41-H44-H45-T45-THTA45-W45
     -TORQ45-NPDT-WS3DT-NP
***** PATH DOES NOT FIT, ADDED TO CARRYOVER FOR NEXT PROCESSOR
***** ORDERED EQUATIONS TO BE PACKED:
  0. WF-DH45-H49-H3
***** TRYING TO INSERT PATH 15
 15. WF
  2. P3-WA31-WF-FAR41-H41-T41-THTA41-PR45Q1-DHQTH4-DH41-H44-H45-T45-THTA45
     -W45-TORQ45-NPDT-NP
***** PATH ADDED
***** UNPACKED PATHS: 3.T3;4.P41;5.P45;6.DHQTH5;8.B1
  9.W45C;10.TORQ41;11.H25;12.WS3;13.WA2;14.DEL2;16.H49;17.H3
  18.WB3;19.H2
***** ORDERED EQUATIONS TO BE PACKED:
  0. DH45-H49-H3
***** TRYING TO INSERT PATH 16
 16. DH45-H49
  2. P3-WA31-WF-FAR41-H41-T41-THTA41-PR45Q1-DHQTH4-DH41-H44-H45-T45-THTA45
     -W45-TORQ45-NPDT-NP-H49
***** PATH DOES NOT FIT, ADDED TO CARRYOVER FOR NEXT PROCESSOR
***** ORDERED EQUATIONS TO BE PACKED:
  0. H49-H3
***** TRYING TO INSERT PATH 16
 16. H49-DH45
  2. P3-WA31-WF-FAR41-H41-T41-THTA41-PR45Q1-DHQTH4-DH41-H44-H45-T45-THTA45
     -W45-TORQ45-NPDT-NP-H49
***** PATH DOES NOT FIT, ADDED TO CARRYOVER FOR NEXT PROCESSOR
```

```
***** ORDERED EQUATIONS TO BE PACKED:
  0. H3
 **** TRYING TO INSERT PATH 17
 17. H3
***** PATH DOES NOT FIT, ADDED TO CARRYOVER FOR NEXT PROCESSOR
***** PACKING CARRYOVER EQUATIONS
***** ORDERED EQUATIONS TO BE PACKED:
  0. WXQ2-B1-TORQ41-H25-WS3DT-WA2-H3-H2
***** TRYING TO INSERT PATH 5
  5. P45-P45DT-WXQ2
***** PATH DOES NOT FIT, ADDED TO CARRYOVER FOR NEXT PROCESSOR
***** ORDERED EQUATIONS TO BE PACKED:
  0. B1-TORQ41-H25-WS3DT-WA2-H3-H2
***** TRYING TO INSERT PATH 8
  8. B1
***** PATH DOES NOT FIT, ADDED TO CARRYOVER FOR NEXT PROCESSOR
***** ORDERED EQUATIONS TO BE PACKED:
  0. TORQ41-H25-WS3DT-WA2-H3-H2
***** TRYING TO INSERT PATH 10
 10. TORQ41
***** PATH DOES NOT FIT, ADDED TO CARRYOVER FOR NEXT PROCESSOR
***** ORDERED EQUATIONS TO BE PACKED:
  0. H25-WS3DT-WA2-H3-H2
***** TRYING TO INSERT PATH 11
 11. H25-T25-T25Q2
***** PATH DOES NOT FIT, ADDED TO CARRYOVER FOR NEXT PROCESSOR
***** ORDERED EQUATIONS TO BE PACKED:
  0. WS3DT-WA2-H3-H2
 **** TRYING TO INSERT PATH 12
 12. WS3-WS3DT
***** PATH DOES NOT FIT, ADDED TO CARRYOVER FOR NEXT PROCESSOR
***** ORDERED EQUATIONS TO BE PACKED:
  0. WA2-H3-H2
***** TRYING TO INSERT PATH 13
 13. WA2
***** PATH DOES NOT FIT, ADDED TO CARRYOVER FOR NEXT PROCESSOR
***** ORDERED EQUATIONS TO BE PACKED:
  0. H3-H2
***** TRYING TO INSERT PATH 17
 17. H3
***** PATH DOES NOT FIT, ADDED TO CARRYOVER FOR NEXT PROCESSOR
***** ORDERED EQUATIONS TO BE PACKED:
  0. H2
***** TRYING TO INSERT PATH 19
 19. H2
***** PATH DOES NOT FIT, ADDED TO CARRYOVER FOR NEXT PROCESSOR

***** PACKING PROCESSOR 3
  3. PS3-PS3Q2-T3Q2-T3C-T3DT-T3
***** RELATIONAL EQUATIONS ARE:
  0. T25Q2
***** CARRYOVER EQUATIONS ARE:
  0. H2-H3-WA2-WS3DT-H25-TORQ41-B1-WXQ2-H3-H49-DH45-WS3DT-TORQ41-W45C-W41
     -PS3
 **** PACKING RELATIONAL EQUATIONS
***** ORDERED EQUATIONS TO BE PACKED:
```

```
  0. T25Q2
***** TRYING TO INSERT PATH 11
 11. T25Q2-T25-H25
  3. PS3-PS3Q2-T25Q2-T3Q2-T3C-T3DT-T3
  3. PS3-PS3Q2-T25Q2-T25-T3Q2-T3C-T3DT-T3
  3. PS3-PS3Q2-T25Q2-T25-H25-T3Q2-T3C-T3DT-T3
***** PATH ADDED
***** UNPACKED PATHS: 4.P41;5.P45;6.DHQTH5;8.B1;9.W45C
  10.TORQ41;12.WS3;13.WA2;14.DEL2;16.H49;17.H3;18.WB3;19.H2
***** PACKING CARRYOVER EQUATIONS
***** ORDERED EQUATIONS TO BE PACKED:
  0. W41-WXQ2-B1-W45C-TORQ41-WS3DT-WA2-DH45-H49-H3-H2
***** TRYING TO INSERT PATH 4
  4. P41-P41DT-W41
  3. PS3-PS3Q2-T25Q2-T25-H25-T3Q2-T3C-T3DT-T3-P41
  3. PS3-PS3Q2-T25Q2-T25-H25-T3Q2-T3C-T3DT-T3-P41DT-P41
  3. PS3-PS3Q2-T25Q2-T25-H25-T3Q2-T3C-T3DT-T3-W41-P41DT-P41
***** PATH ADDED
***** RELATIONAL EQUATIONS ARE:
  0. TORQ41-P45DT
***** UNPACKED PATHS: 5.P45;6.DHQTH5;8.B1;9.W45C;10.TORQ41
  12.WS3;13.WA2;14.DEL2;16.H49;17.H3;18.WB3;19.H2
***** ORDERED EQUATIONS TO BE PACKED:
  0. P45DT-TORQ41
***** NOT PATH ENDPOINT:P45DT
***** ORDERED EQUATIONS TO BE PACKED:
  0. TORQ41
***** TRYING TO INSERT PATH 10
 10. TORQ41
  3. PS3-PS3Q2-T25Q2-T25-H25-T3Q2-T3C-T3DT-T3-W41-TORQ41-P41DT-P41
***** PATH ADDED
***** UNPACKED PATHS: 5.P45;6.DHQTH5;8.B1;9.W45C;12.WS3
  13.WA2;14.DEL2;16.H49;17.H3;18.WB3;19.H2
***** ORDERED EQUATIONS TO BE PACKED:
  0. WXQ2-B1-W45C-WS3DT-WA2-DH45-H49-H3-H2
***** TRYING TO INSERT PATH 5
  5. P45-P45DT-WXQ2
  3. PS3-PS3Q2-T25Q2-T25-H25-T3Q2-T3C-T3DT-T3-W41-TORQ41-P41DT-P41-P45
  3. PS3-PS3Q2-T25Q2-T25-H25-T3Q2-T3C-T3DT-T3-W41-TORQ41-P41DT-P41-P45DT
     -P45
  3. PS3-PS3Q2-T25Q2-T25-H25-T3Q2-T3C-T3DT-T3-W41-TORQ41-P41DT-P41-WXQ2-P45DT
     -P45
***** PATH ADDED
***** RELATIONAL EQUATIONS ARE:
  0. WA2-WB3
***** UNPACKED PATHS: 6.DHQTH5;8.B1;9.W45C;12.WS3;13.WA2
  14.DEL2;16.H49;17.H3;18.WB3;19.H2
***** ORDERED EQUATIONS TO BE PACKED:
  0. WA2-WB3
***** PACKING DELAYED, NOT CRITICAL:WA2
***** ORDERED EQUATIONS TO BE PACKED:
  0. WB3
***** TRYING TO INSERT PATH 18
 18. WB3
  3. PS3-PS3Q2-T25Q2-T25-H25-T3Q2-T3C-T3DT-T3-W41-TORQ41-P41DT-P41-WXQ2-WB3
     -P45DT-P45
```

```
***** PATH ADDED
***** RELATIONAL EQUATIONS ARE:
  0. WA2-WS3DT
***** UNPACKED PATHS: 6.DHQTH5;8.B1;9.W45C;12.WS3;13.WA2
  14.DEL2;16.H49;17.H3;19.H2
***** ORDERED EQUATIONS TO BE PACKED:
  0. WS3DT-WA2
***** TRYING TO INSERT PATH 12
 12. WS3DT-WS3
  3. PS3-PS3Q2-T25Q2-T25-H25-T3Q2-T3C-T3DT-T3-W41-TORQ41-P41DT-P41-WXQ2-WB3
     -P45DT-WS3DT-P45
***** PATH DOES NOT FIT, ADDED TO CARRYOVER FOR NEXT PROCESSOR
***** ORDERED EQUATIONS TO BE PACKED:
  0. WA2
***** PACKING DELAYED, NOT CRITICAL:WA2
***** ORDERED EQUATIONS TO BE PACKED:
  0. B1-W45C-WS3DT-WA2-DH45-H49-H3-H2
***** TRYING TO INSERT PATH 8
  8. B1
***** PATH DOES NOT FIT, ADDED TO CARRYOVER FOR NEXT PROCESSOR
***** ORDERED EQUATIONS TO BE PACKED:
  0. W45C-WS3DT-WA2-DH45-H49-H3-H2
***** TRYING TO INSERT PATH 9
  9. W45C
***** PATH DOES NOT FIT, ADDED TO CARRYOVER FOR NEXT PROCESSOR
***** ORDERED EQUATIONS TO BE PACKED:
  0. WS3DT-WA2-DH45-H49-H3-H2
***** TRYING TO INSERT PATH 12
 12. WS3-WS3DT
  3. PS3-PS3Q2-T25Q2-T25-H25-T3Q2-T3C-T3DT-T3-W41-TORQ41-P41DT-P41-WXQ2-WB3
     -P45DT-WS3DT-P45
***** PATH DOES NOT FIT, ADDED TO CARRYOVER FOR NEXT PROCESSOR
***** ORDERED EQUATIONS TO BE PACKED:
  0. WA2-DH45-H49-H3-H2
***** TRYING TO INSERT PATH 13
 13. WA2
***** PATH DOES NOT FIT, ADDED TO CARRYOVER FOR NEXT PROCESSOR
***** ORDERED EQUATIONS TO BE PACKED:
  0. DH45-H49-H3-H2
***** TRYING TO INSERT PATH 16
 16. H49-DH45
  3. PS3-PS3Q2-T25Q2-T25-H25-T3Q2-T3C-T3DT-T3-W41-TORQ41-P41DT-P41-WXQ2-WB3
     -P45DT-P45-H49
***** PATH DOES NOT FIT, ADDED TO CARRYOVER FOR NEXT PROCESSOR
***** ORDERED EQUATIONS TO BE PACKED:
  0. H49-H3-H2
***** TRYING TO INSERT PATH 16
 16. H49-DH45
  3. PS3-PS3Q2-T25Q2-T25-H25-T3Q2-T3C-T3DT-T3-W41-TORQ41-P41DT-P41-WXQ2-WB3
     -P45DT-P45-H49
***** PATH DOES NOT FIT, ADDED TO CARRYOVER FOR NEXT PROCESSOR
***** ORDERED EQUATIONS TO BE PACKED:
  0. H3-H2
***** TRYING TO INSERT PATH 17
 17. H3
  3. H3-PS3-PS3Q2-T25Q2-T25-H25-T3Q2-T3C-T3DT-T3-W41-TORQ41-P41DT-P41-WXQ2
```

```
     -WB3-P45DT-P45
***** PATH ADDED
 ***** UNPACKED PATHS: 6.DHQTH5;8.B1;9.W45C;12.WS3;13.WA2
  14.DEL2;16.H49;19.H2
***** ORDERED EQUATIONS TO BE PACKED:
  0. H2
***** TRYING TO INSERT PATH 19
 19. H2
  3. H3-H2-PS3-PS3Q2-T25Q2-T25-H25-T3Q2-T3C-T3DT-T3-W41-TORQ41-P41DT-P41
     -WXQ2-WB3-P45DT-P45
***** PATH ADDED
***** UNPACKED PATHS: 6.DHQTH5;8.B1;9.W45C;12.WS3;13.WA2
  14.DEL2;16.H49

***** PACKING PROCESSOR 4
  4. PR49Q5-DHQTH5
***** RELATIONAL EQUATIONS ARE:
  0. W45C-DH45
***** CARRYOVER EQUATIONS ARE:
  0. H49-DH45-WA2-WS3DT-W45C-B1-WS3DT
***** PACKING RELATIONAL EQUATIONS
***** ORDERED EQUATIONS TO BE PACKED:
  0. W45C-DH45
***** TRYING TO INSERT PATH 9
  9. W45C
  4. PR49Q5-W45C-DHQTH5
***** PATH ADDED
***** UNPACKED PATHS: 8.B1;12.WS3;13.WA2;14.DEL2;16.H49
 ***** ORDERED EQUATIONS TO BE PACKED:
  0. DH45
***** TRYING TO INSERT PATH 16
 16. DH45-H49
  4. PR49Q5-W45C-DHQTH5-DH45
  4. PR49Q5-W45C-DHQTH5-DH45-H49
***** PATH ADDED
***** UNPACKED PATHS: 8.B1;12.WS3;13.WA2;14.DEL2
***** PACKING CARRYOVER EQUATIONS
***** ORDERED EQUATIONS TO BE PACKED:
  0. B1-WS3DT-WA2
***** TRYING TO INSERT PATH 8
  8. B1
  4. PR49Q5-W45C-DHQTH5-B1-DH45-H49
***** PATH ADDED
***** UNPACKED PATHS: 12.WS3;13.WA2;14.DEL2
***** ORDERED EQUATIONS TO BE PACKED:
  0. WS3DT-WA2
***** TRYING TO INSERT PATH 12
 12. WS3-WS3DT
  4. PR49Q5-W45C-DHQTH5-B1-DH45-H49-WS3
  4. PR49Q5-W45C-DHQTH5-B1-DH45-H49-WS3DT-WS3
***** PATH ADDED
***** UNPACKED PATHS: 13.WA2;14.DEL2
***** ORDERED EQUATIONS TO BE PACKED:
  0. WA2
 ***** TRYING TO INSERT PATH 13
 13. WA2
```

```
  4. PR49Q5-W45C-DHQTH5-B1-DH45-WA2-H49-WS3DT-WS3
***** PATH ADDED
 **** RELATIONAL EQUATIONS ARE:
  0. DEL2
***** UNPACKED PATHS: 14.DEL2
***** ORDERED EQUATIONS TO BE PACKED:
  0. DEL2
***** TRYING TO INSERT PATH 14
 14. DEL2
  4. PR49Q5-W45C-DHQTH5-B1-DEL2-DH45-WA2-H49-WS3DT-WS3
***** PATH ADDED
***** UNPACKED PATHS: NONE
* SMALLEST PATH EXECUTION TIME:3042, UNUSED TIME ON OTHER PATHS:186
PACKING SEQUENCE COMPLETE
```

```
* * * * * * * * * * * *
  PROCESSOR  1          *
* UTILIZATION:100%      *
* * * * * * * * * * * *
```

<-------- PROCESSOR EQUATION ---------->				<---- SENT TO ---->		<------ NEEDS ------>	
RESULT	CALC	START	END	MAX	PR EQUATION	REQ	PR EQUATION	AVAIL
RTTH2	770	0	770	770	4 WA2	3706		
NGC	434	770	1204	1204				
PCNGC	198	1204	1402	1402	4 B1	1438		
WA2C	2136	1402	3538	3538	3 WXQ2 4 WA2	4138 3706	3 PS3Q2	594
B2	610	3538	4148	4148				
WB25	122	4148	4270	4270			4 B1 4 WA2	2036 3998
WA3	30	4270	4300	4300	4 WS3DT	4866	4 WA2	3998
TORQC	594	4300	4894	4894			3 H3 3 H2 3 H25 4 WA2	142 268 960 3998
NGDT	566	4894	5460	5460			3 TORQ41	3620
NG	206	5460	5666	5666				

```
* * * * * * * * * * * *
* PROCESSOR  2          *
* UTILIZATION: 98%      *
* * * * * * * * * * * *
```

<-------- PROCESSOR EQUATION ---------->				<---- SENT TO ---->		<------ NEEDS ------>	
RESULT	CALC	START	END	MAX	PR EQUATION	REQ	PR EQUATION	AVAIL
P3	242	0	242	304	3 PS3	268		
WA31	970	242	1212	1274	3 P41DT 4 WS3DT	3620 4866		
WF	214	1212	1426	1488				
FAR41	224	1426	1650	1712				
H41	336	1650	1986	2048			3 H3	142
T41	122	1986	2108	2170	3 P41DT	3620		
THTA41	122	2108	2230	2292	3 W41	2392		

PR45Q1	230	2230	2460	2522				
DHQTH4	598	2460	3058	3120				
DH41	118	3058	3176	3238	3	TORQ41	3230	
H44	54	3176	3230	3292				
H45	114	3230	3344	3406	4	H49	3998	
T45	122	3344	3466	3528	3	P45DT	4866	
THTA45	122	3466	3588	3650	4	DH45	3588	
W45	854	3588	4442	4504	3	P45DT	4866	4 W45C 828
TORQ45	390	4442	4832	4894				4 DH45 3706
NPDT	566	4832	5398	5460				
NP	206	5398	5604	5666				

```
* * * * * * * * * * * *
* PROCESSOR  3         *
* UTILIZATION: 97%     *
* * * * * * * * * * * *
```

PROCESSOR EQUATION RESULT	CALC	START	END	MAX	SENT TO PR EQUATION	REQ	NEEDS PR EQUATION	AVAIL
H3	142	0	142	266	1 TORQC 2 H41	4300 1650		
H2	126	142	268	392	1 TORQC	4300		
PS3	122	268	390	514			2 P3	242
PS3Q2	204	390	594	718	1 WA2C	1402		
T25Q2	134	594	728	852				
T25	118	728	846	970				
H25	114	846	960	1084	1 TORQC	4300		
T3Q2	598	960	1558	1682				
T3C	134	1558	1692	1816				
T3DT	494	1692	2186	2310				
T3	206	2186	2392	2516				
.41	838	2392	3230	3354			2 THTA41	2230

TORQ41	390	3230	3620	3744	1 NGDT	4894	2 DH41	3176
41DT	312	3620	3932	4056			2 WA31	1212
							2 T41	2108
P41	206	3932	4138	4262				
WXQ2	598	4138	4736	4860			1 WA2C	3538
WB3	130	4736	4866	4990	4 WS3DT	4866	4 WA2	3998
P45DT	470	4866	5336	5460			2 T45	3466
							2 W45	4442
							4 WA2	3998
P45	206	5336	5542	5666				

* * * * * * * * * * * *
* PROCESSOR 4 *
* UTILIZATION: 53% *
* * * * * * * * * * * *

<-------- PROCESSOR EQUATION -------->					<----- SENT TO ---->		<------ NEEDS ------>	
RESULT	CALC	START	END	MAX	PR EQUATION	REQ	PR EQUATION	AVAIL
PR49Q5	230	0	230	1714				
W45C	598	230	828	2312	2 W45	3588		
DHQTH5	610	828	1438	2922				
B1	598	1438	2036	3520	1 WB25	4148	1 PCNQC	1402
DEL2	218	2036	2254	3738				
DH45	118	3588	3706	3856	2 TORQ45	4442	2 THTA45	3588
WA2	292	3706	3998	4148	1 WB25	4148	1 RTTH2	770
					1 WA3	4270	1 WA2C	3538
					1 TORQC	4300		
					3 WB3	4736		
					3 P45DT	4866		
H49	42	3998	4040	5330			2 H45	3344
WS3DT	130	4866	4996	5460			1 WA3	4300
							2 WA31	1212
							3 WB3	4866
WS3	206	4996	5202	5666				

REFERENCES

1 Blech, R.A.; and Arpasi, D.J.: Hardware for a Real-Time Multiprocessor Simulator. NASA TM-83805, 1984.

2. Arpasi, D.J.: RTMPL - A Structured Programming and Documentation Utility for Real-Time Multiprocessor Simulations. NASA TM-83606, 1985.

3. Arpasi, Dale J.: Real-Time Multiprocessor Programming Language, (RTMPL) - Users Manual. NASA TP-2422, 1985.

4. Cole, G.L.: Operating System for a Real-Time Multiprocessor Propulsion System Simulator. NASA TM-83605, 1985.

TABLE I. - RTMPL - GENERATED INFORMATION

Label	Operation	Result	Arguments	Calculation time
S$12	LOAD	R1	z	8
	ADD	R1	R1,y	16
	STORE	X	R1	8

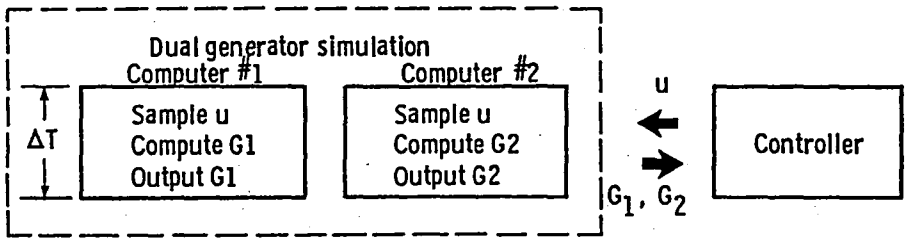
TABLE II. - PARTITIONING INFORMATION

Equation label	Dependent arguments	Independent arguments	Calculation time	Can start	Can end	Must end[a]
X_1	(none)	U, X_5	32	0	32	64
X_2	(none)	X_5	64	0	64	64
X_3	X_2	(none)	32	64	96	112
X_4	X_1, X_2	(none)	48	64	112	112
X_5	X_3, X_4	X_5	48	112	160	160

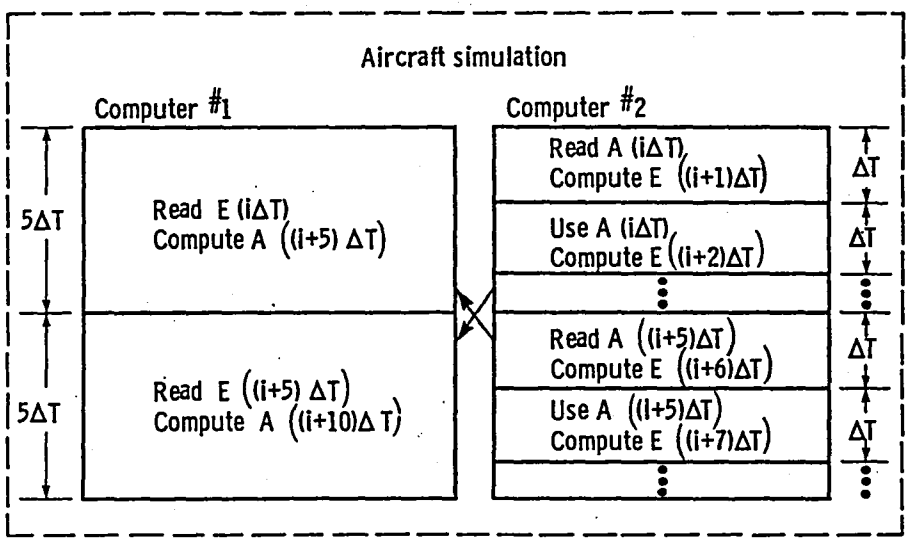
[a]Information used for packing only.

TABLE III. - PACKING ALGORITHM RESULTS FOR TURBOJET MODEL

Update time	Processors required	Processor percent utilization			
		P number 1	P number 2	P number 3	P number 4
5666	4	100	98	97	53
10000	2	99	98	--	--
19568	1	100	--	--	--

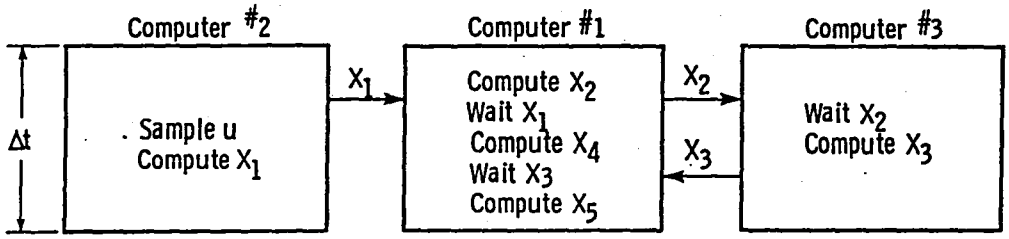


(a) Decoupled computational paths.

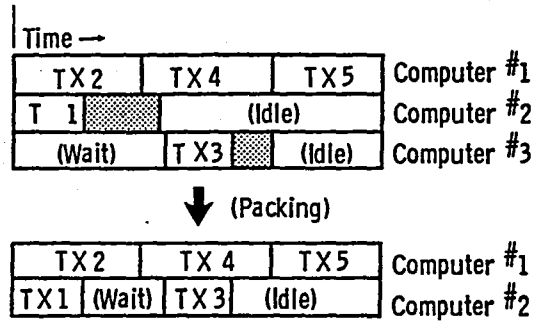


(b) Loosely coupled computational paths.

Figure 1. - Physical parallelism.



(a) Closely coupled paths.



(b) Packing.

Figure 2. - Partitioning and packing closely coupled equations.

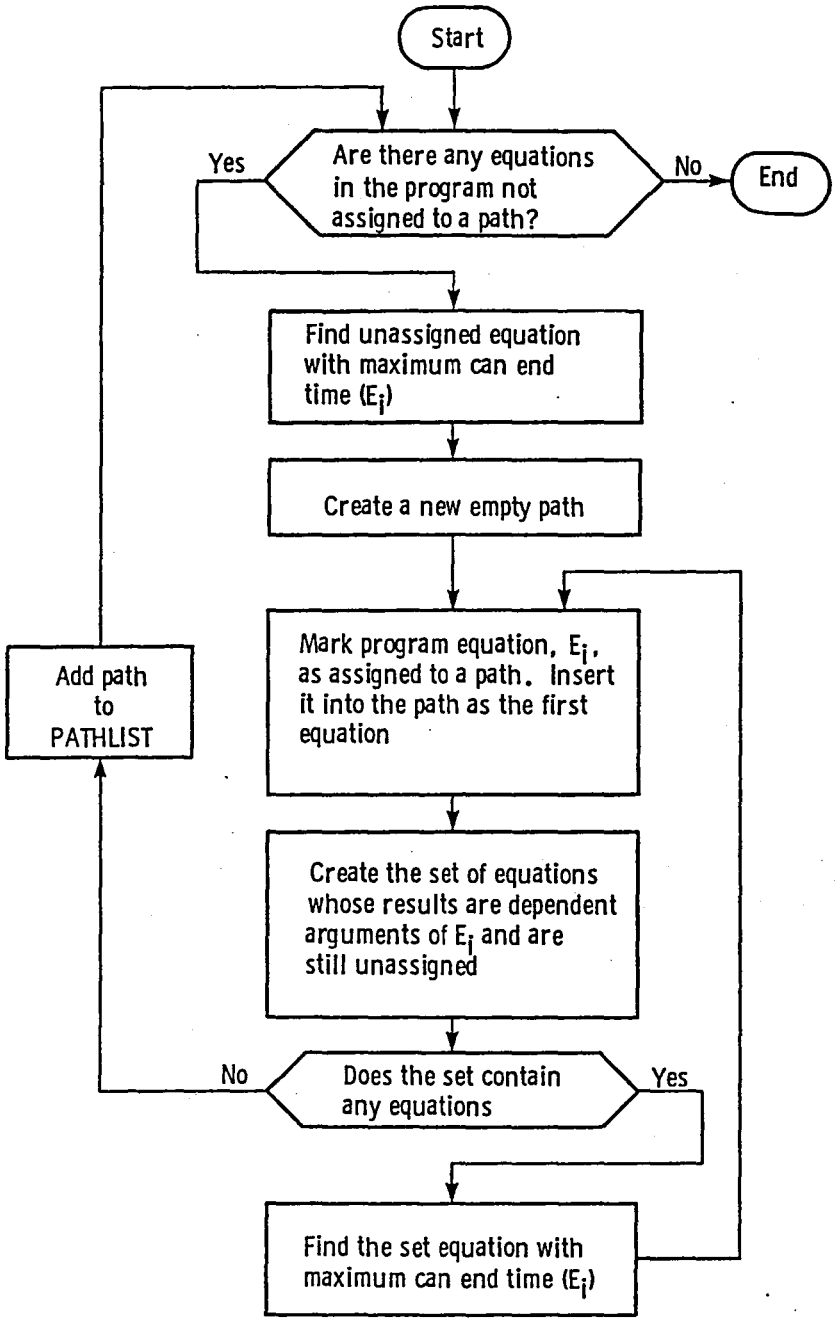
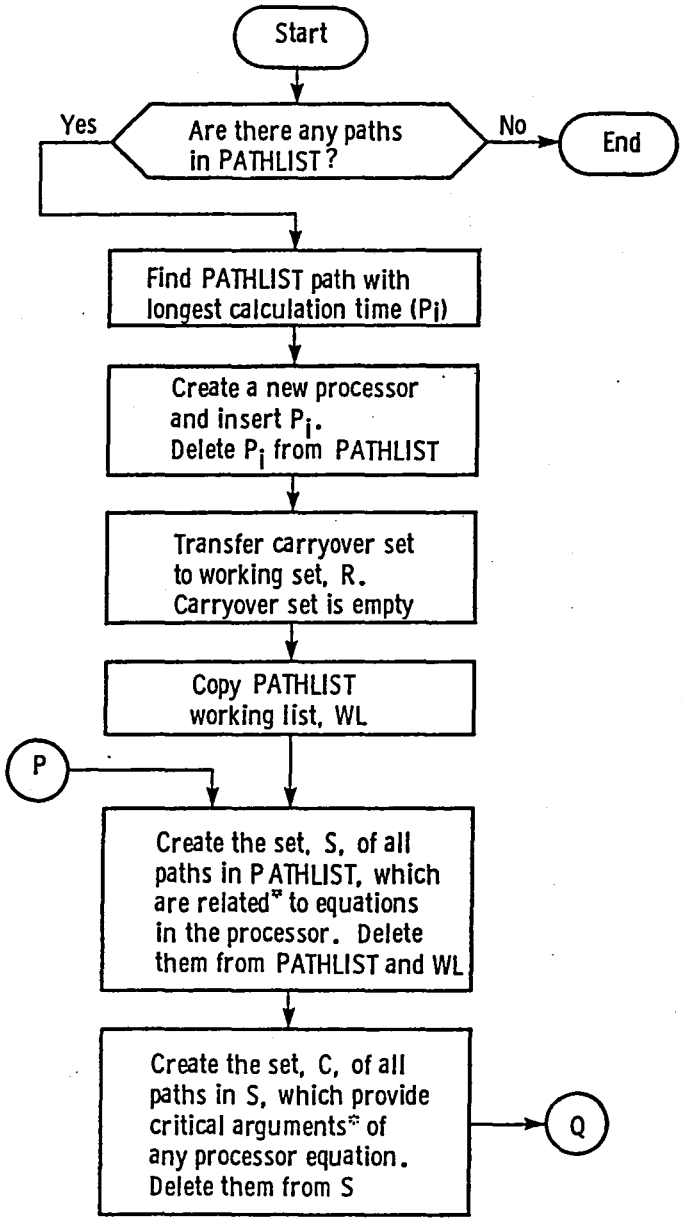


Figure 3. - Path identification Algorithm.



*See explanation in test.

Figure 4. - Packing Algorithm.

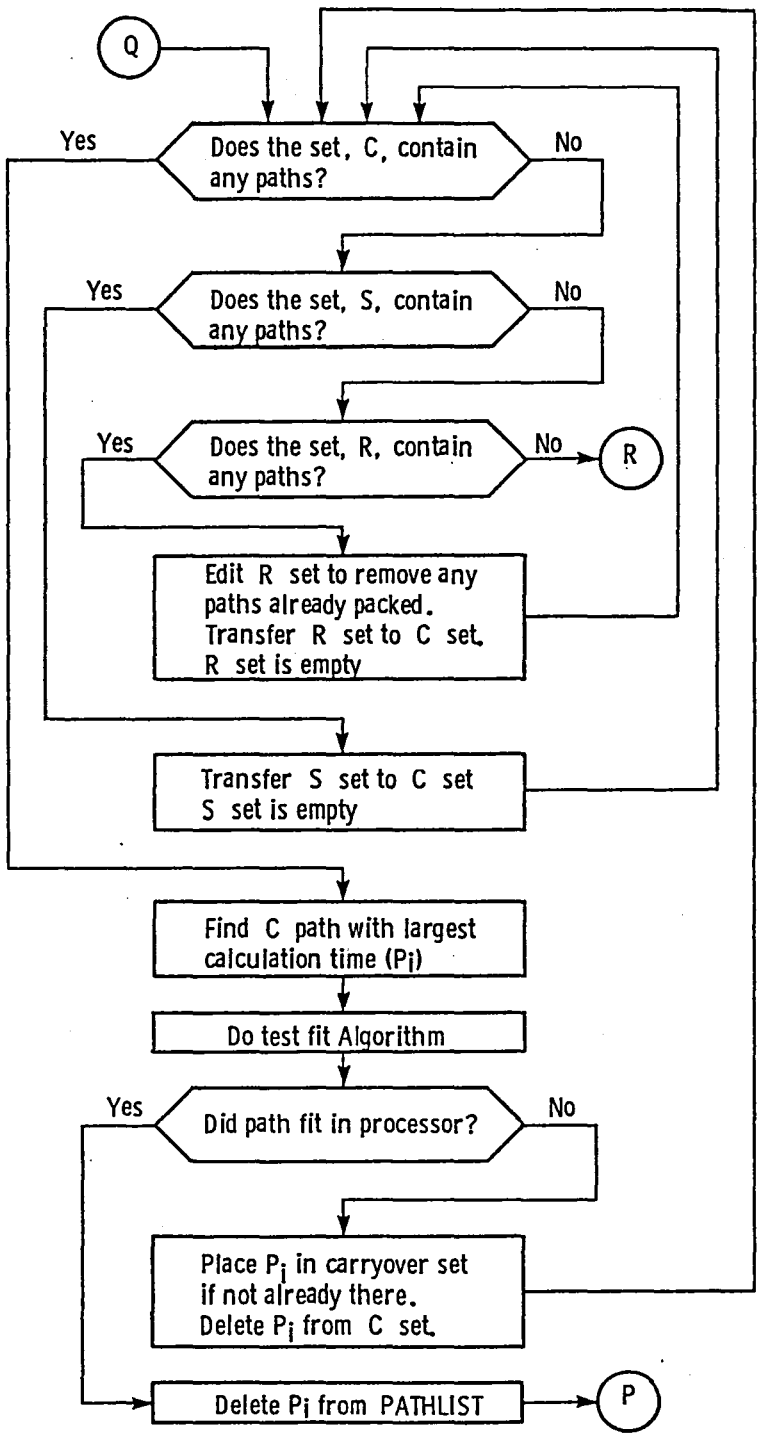


Figure 4. - Continued.

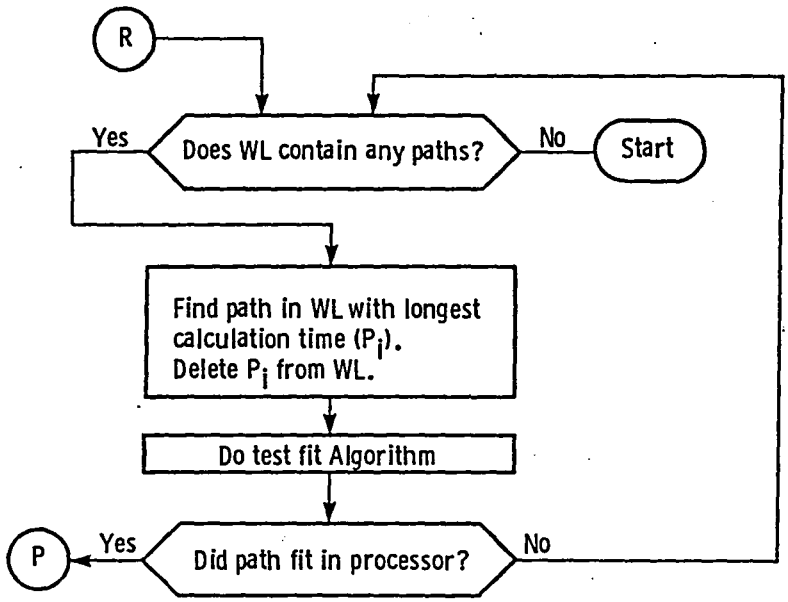
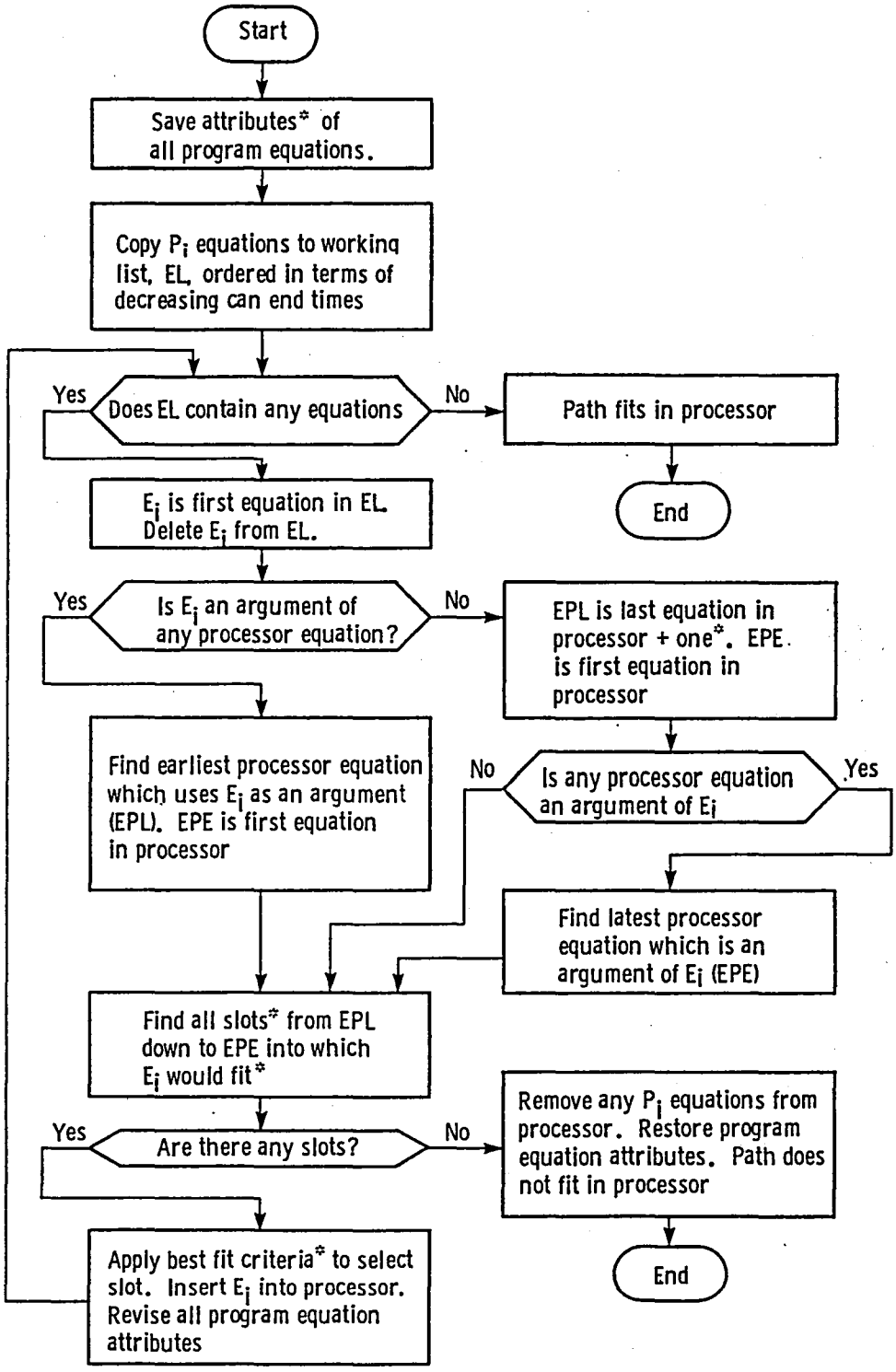


Figure 4. - Concluded.



*See explanation in text

Figure 5. - Test fit Algorithm.

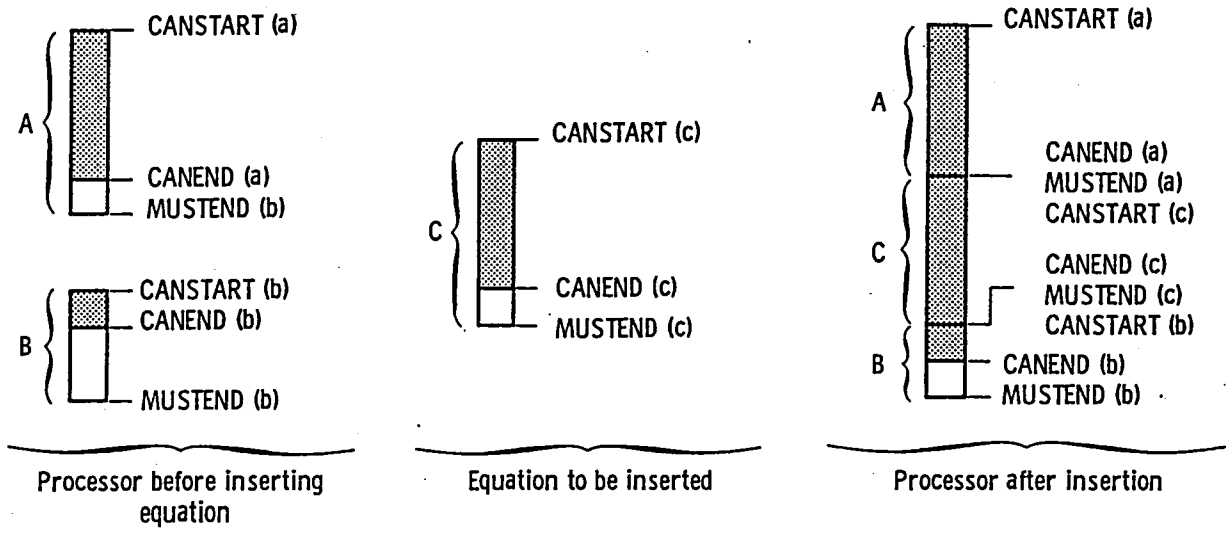


Figure 6. - The affect of insertion on equation attributes.

End of Document